网络空间安全创新创业实践

ECDSA pitfalls

实验报告

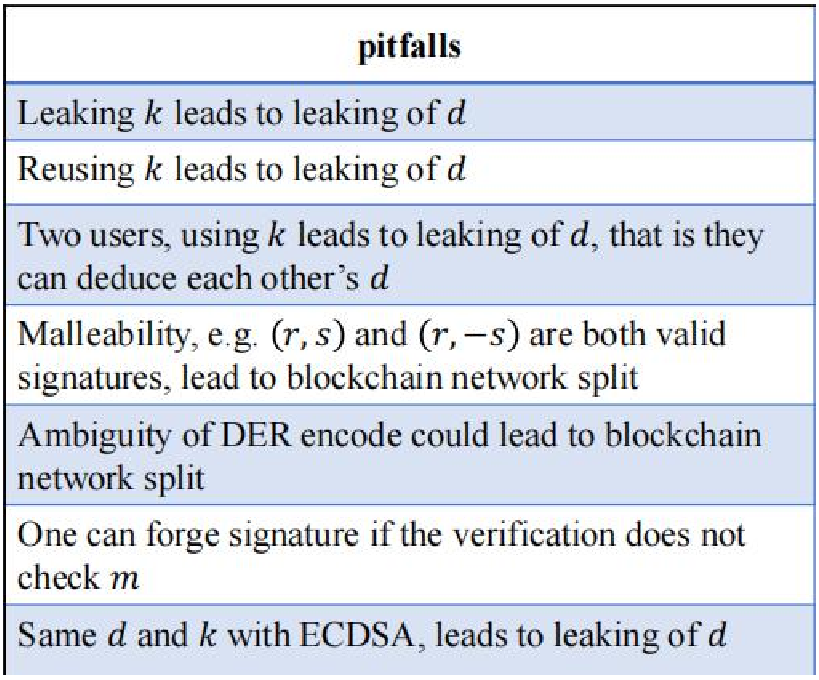
王祥宇---202000460053

项目时间：2022 年 7 月 18 日

目录

一、项目任务

Verify the below pitfalls with proof-of-concept code :



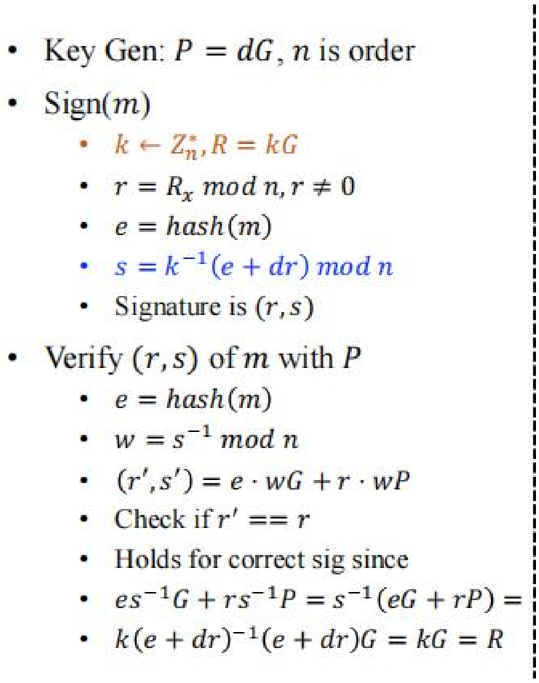
pitfalls
Leaking k leads to leaking of d
Reusing k leads to leaking of d
Two users, using k leads to leaking of d, that is they can deduce each other's d
Malleability, e.g. (r, s) and $(r, -s)$ are both valid signatures, lead to blockchain network split
Ambiguity of DER encode could lead to blockchain network split
One can forge signature if the verification does not check m
Same d and k with ECDSA, leads to leaking of d

在实现上述证明之前，需要完成一些基本的运算，比如说求最大公因子 gcd、求乘法逆元 Extended_Euclidean、椭圆曲线上的点的加法 Add 和数乘 Multiply 运算，这些均在代码最前面实现。由于思路比较简单且其余课程也多次实现故在此不在赘述思路。

二、实验过程及思路

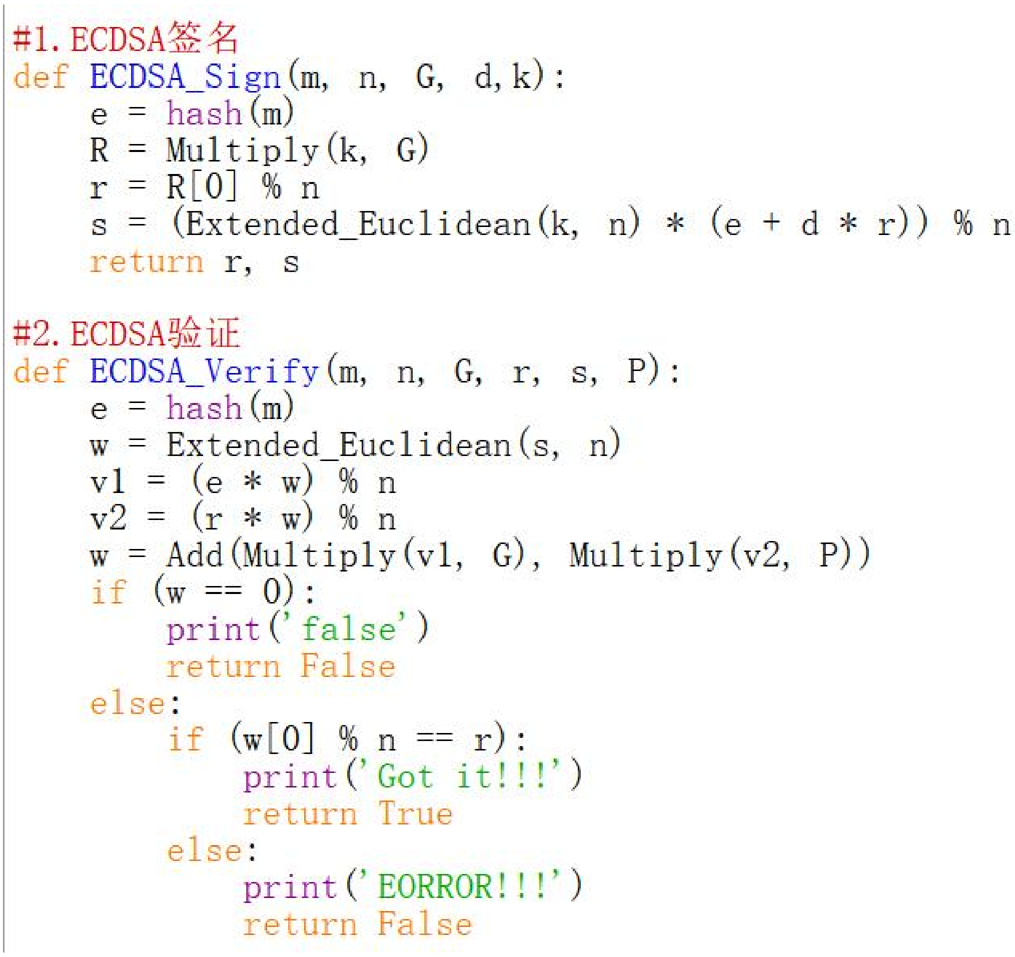
任务一：ECDSA 签名和 ECDSA 验证

ECDSA 签名和验证的思路上课已经讲过，并且给出了伪代码实现，如下图：



- Key Gen: $P = dG$, n is order
- Sign(m)
  - $k \leftarrow Z_n^*, R = kG$
  - $r = R_x \ mod\ n, r \neq 0$
  - $e = hash(m)$
  - $s = k^{-1}(e + dr)\ mod\ n$
  - Signature is (r, s)
- Verify (r, s) of m with P
  - $e = hash(m)$
  - $w = s^{-1}\ mod\ n$
  - $(r', s') = e \cdot wG + r \cdot wP$
  - Check if $r' == r$
  - Holds for correct sig since
  - $es^{-1}G + rs^{-1}P = s^{-1}(eG + rP) =$
  - $k(e + dr)^{-1}(e + dr)G = kG = R$

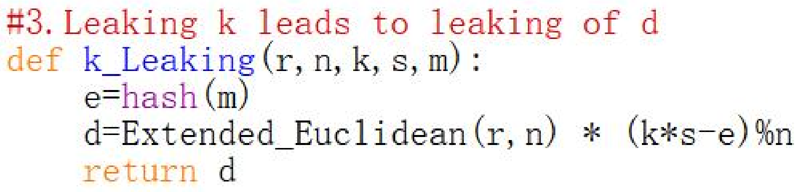
所以直接实现即可。具体实现代码如下图所示:



```
#1.ECDSA签名
def ECDSA_Sign(m, n, G, d,k):
    e = hash(m)
    R = Multiply(k, G)
    r = R[0] % n
    s = (Extended_Euclidean(k, n) * (e + d * r)) % n
    return r, s

#2.ECDSA验证
def ECDSA_Verify(m, n, G, r, s, P):
    e = hash(m)
    w = Extended_Euclidean(s, n)
    v1 = (e * w) % n
    v2 = (r * w) % n
    w = Add(Multiply(v1, G), Multiply(v2, P))
    if (w == 0):
        print('false')
        return False
    else:
        if (w[0] % n == r):
            print('Got it!!!')
            return True
        else:
            print('EORROR!!!')
            return False
```

任务二：Leaking *k* leads to leaking of *d*

根据公式 $s = k^{-1}(e + dr)\ mod\ n$ 可以直接得出 d=(s*k-e)*r(-1) mod n，而其中 e,k,r,s 敌手均可知，故随机数 k 的泄露会导致 d 的泄露。具体实现代码如下图：



```
#3.Leaking k leads to leaking of d
def k_Leaking(r,n,k,s,m):
    e=hash(m)
    d=Extended_Euclidean(r,n) * (k*s-e)%n
    return d
```

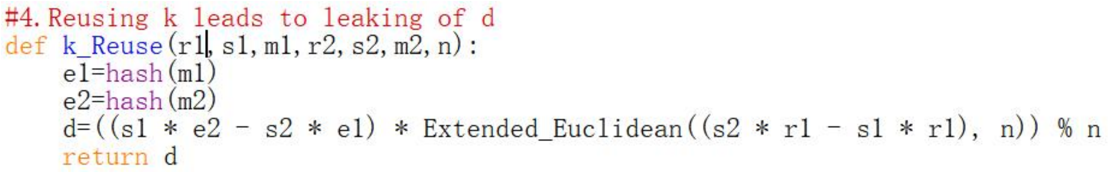
任务三：Reusing *k* leads to leaking of *d*

两次签名使用了同样的随机数，，并且对于相同的 G 和 P，r 也相同且已知，所以联立 s 的方程可得：

$$s1 = k^{-1}(e1+dr)\ \mathrm{mod}\ n$$

$$s2 = k^{-1}(e2+dr)\ \mathrm{mod}\ n$$

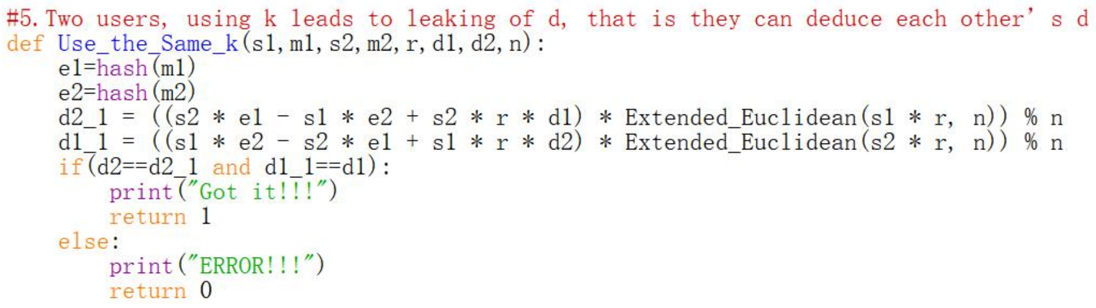
两个方程，两个未知数 k 和 d，故可以求解出 d。具体实现代码如下图：



```
#4.Reusing k leads to leaking of d
def k_Reuse(r1,s1,m1,r2,s2,m2,n):
    e1=hash(m1)
    e2=hash(m2)
    d=((s1 * e2 - s2 * e1) * Extended_Euclidean((s2 * r1 - s1 * r1), n)) % n
    return d
```

任务四：Two users, using *k* leads to leaking of *d*, that is they can deduce each other's *d*

该任务和任务二泄露 k 类似。具体实现代码入下图：



```
#5.Two users, using k leads to leaking of d, that is they can deduce each other’s d
def Use_the_Same_k(s1,m1,s2,m2,r,d1,d2,n):
    e1=hash(m1)
    e2=hash(m2)
    d2_1 = ((s2 * e1 - s1 * e2 + s2 * r * d1) * Extended_Euclidean(s1 * r, n)) % n
    d1_1 = ((s1 * e2 - s2 * e1 + s1 * r * d2) * Extended_Euclidean(s2 * r, n)) % n
    if(d2==d2_1 and d1_1==d1):
        print("Got it!!!")
        return 1
    else:
        print("ERROR!!!")
        return 0
```

任务五：Malleability, e.g. (r,s) and (r,-s) are both valid signatures, lead to blockchain network split

从验证公式来看，（r,s）和（r,-s）均能通过验证，如下图所示：

$$e\cdot(-s)^{-1}G+r\cdot(-s)^{-1}P=-(e\cdot s^{-1}G+r\cdot s^{-1}P)=(x',-y'),\ r=x'mod\ p$$

故带入验证测试仍然通过。

任务六：One can forge signature if the verification does not check *m*

由验证算法我们可得：

$$s^{-1}(e*G+rP)\ \mathrm{mod}\ n = (r,s)$$

所以给定 u 和 v，使得 u*G+v*P mod n=(r, s)。此时构造签名（r1, s1）满足下式：

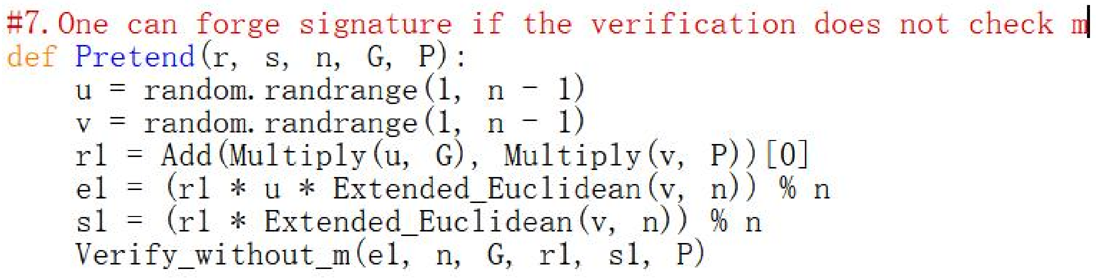
$$r1 = r\ \mathrm{mod}\ n$$

$$s1 = r1 * v^{-1}\ \mathrm{mod}\ n$$

构造假的 hash 值

$e' = r1*u*v^{-1} \bmod n$

所以，这个新的签名对同样对 k 和 d 有效。具体实现代码如下图所示：

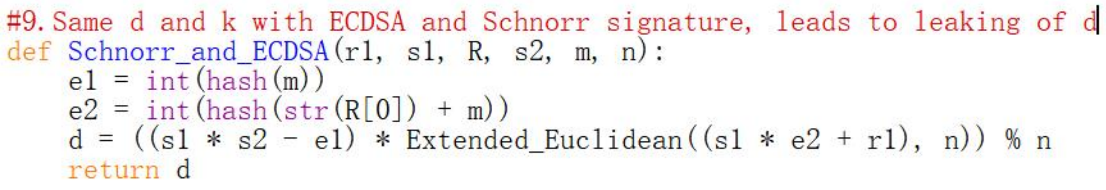


```
#7.One can forge signature if the verification does not check m
def Pretend(r, s, n, G, P):
    u = random.randrange(1, n - 1)
    v = random.randrange(1, n - 1)
    r1 = Add(Multiply(u, G), Multiply(v, P))[0]
    e1 = (r1 * u * Extended_Euclidean(v, n)) % n
    s1 = (r1 * Extended_Euclidean(v, n)) % n
    Verify_without_m(e1, n, G, r1, s1, P)
```

任务七：Same *d* and *k* with ECDSA and Schnorr signature, leads to leaking of *d*

当 d 和 k 相同时，根据 s1 和 s2 的计算公式，消除 k，从而接的密钥 d。

具体实现代码如下图：

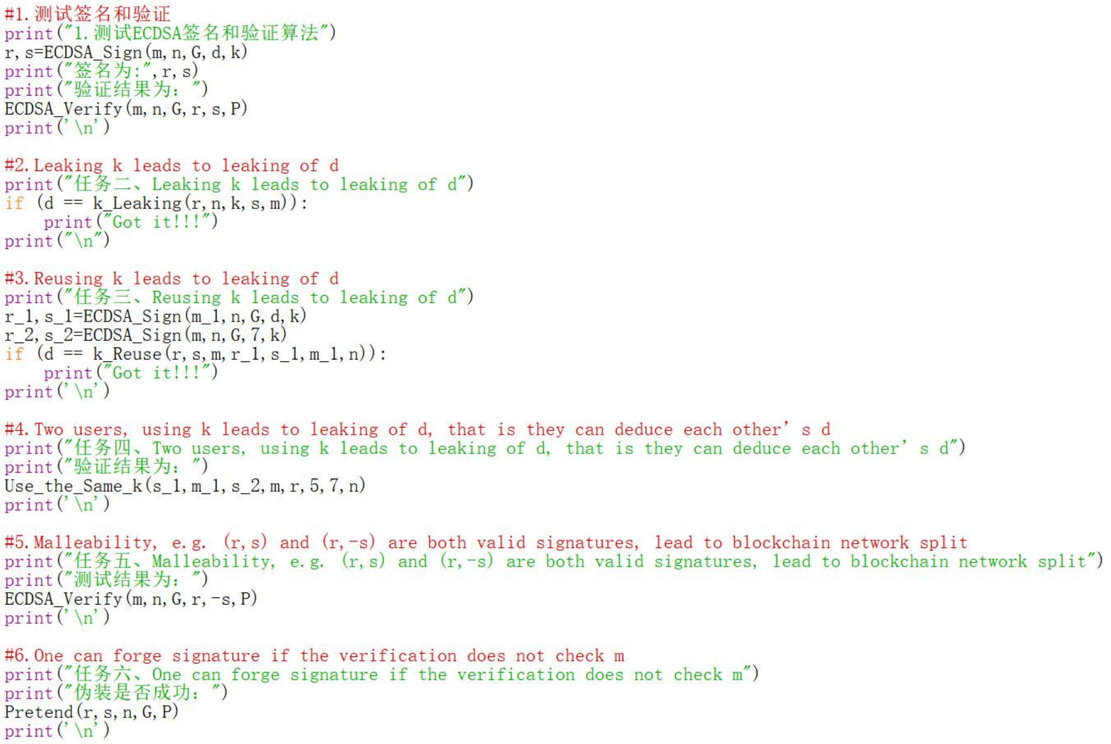


```
#9.Same d and k with ECDSA and Schnorr signature, leads to leaking of d
def Schnorr_and_ECDSA(r1, s1, R, s2, m, n):
    e1 = int(hash(m))
    e2 = int(hash(str(R[0]) + m))
    d = ((s1 * s2 - e1) * Extended_Euclidean((s1 * e2 + r1), n)) % n
    return d
```

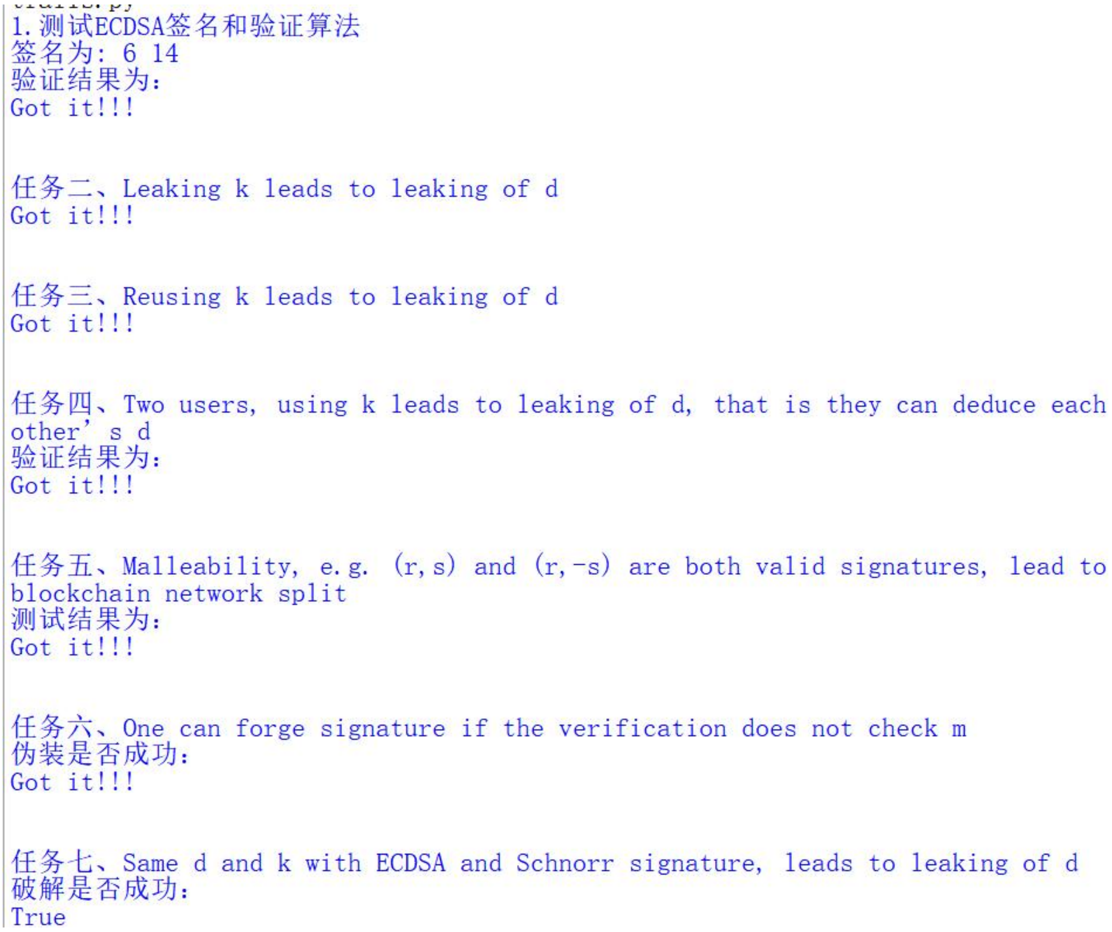
三、项目测试

利用上面的任务函数进行测试，代码和结果如下图：

ECDSA pitfalls 实验报告



```
#1.测试签名和验证
print("1.测试ECDSA签名和验证算法")
r,s=ECDSA_Sign(m,n,G,d,k)
print("签名为:",r,s)
print("验证结果为: ")
ECDSA_Verify(m,n,G,r,s,P)
print('\n')

#2.Leaking k leads to leaking of d
print("任务二、Leaking k leads to leaking of d")
if (d == k_Leaking(r,n,k,s,m)):
    print("Got it!!!")
print("\n")

#3.Reusing k leads to leaking of d
print("任务三、Reusing k leads to leaking of d")
r_1,s_1=ECDSA_Sign(m_1,n,G,d,k)
r_2,s_2=ECDSA_Sign(m,n,G,7,k)
if (d == k_Reuse(r,s,m,r_1,s_1,m_1,n)):
    print("Got it!!!")
print('\n')

#4.Two users, using k leads to leaking of d, that is they can deduce each other's d
print("任务四、Two users, using k leads to leaking of d, that is they can deduce each other's d")
print("验证结果为: ")
Use_the_Same_k(s_1,m_1,s_2,m,r,5,7,n)
print('\n')

#5.Malleability, e.g. (r,s) and (r,-s) are both valid signatures, lead to blockchain network split
print("任务五、Malleability, e.g. (r,s) and (r,-s) are both valid signatures, lead to blockchain network split")
print("测试结果为: ")
ECDSA_Verify(m,n,G,r,-s,P)
print('\n')

#6.One can forge signature if the verification does not check m
print("任务六、One can forge signature if the verification does not check m")
print("伪装是否成功: ")
Pretend(r,s,n,G,P)
print('\n')
```



```
1.测试ECDSA签名和验证算法
签名为: 6 14
验证结果为:
Got it!!!


任务二、Leaking k leads to leaking of d
Got it!!!


任务三、Reusing k leads to leaking of d
Got it!!!


任务四、Two users, using k leads to leaking of d, that is they can deduce each
other's d
验证结果为:
Got it!!!


任务五、Malleability, e.g. (r,s) and (r,-s) are both valid signatures, lead to
blockchain network split
测试结果为:
Got it!!!


任务六、One can forge signature if the verification does not check m
伪装是否成功:
Got it!!!


任务七、Same d and k with ECDSA and Schnorr signature, leads to leaking of d
破解是否成功:
True
```

四、实验反思与总结

通过此项目的几个任务证明，了解并实现了 ECDSA 的相关内容，对 ECDSA 的安全性有了更加深入的了解。在密码学引论这么课上只学到了一般的 DSA 签名算法，通过这个项目了解了在椭圆曲线上的 DSA 的相关实现，为以后的密码学习奠定了基础。